第五章练习题

1、填空题

（1）C 语言中实现循环结构的控制语句有__for__语句、__wlihe__语句和__do wlihe__语句。

（2）当循环体内遇到__break__、__continue__或__return__语句时，将退出循环。

（3）do…while 语句和 while 语句的区别在于__do 先执行一次，再判断是否循环__。

（ 4） break 语句在循环体中的作用是 __直接跳出循环__，continue 语句在循环体中的作用是__跳出当前循环，并执行下一次循环__。

（5）如果循环次数在执行循环体之前就已确定，一般用__for__循环；如果循环次数是有循环体的执行情况确定的，一般用__wlihe__循环或者__do wlihe__循环。当循环体至少执行一次时，用__do wlihe__循环，反之，如果循环体可能一次也不执行，选用__wlihe__循环。

2、选择题

（1）若 i 为整型变量，则以下循环执行次数是（ D ）

```
for (i = 2; i != 0;) printf (”%d”, i--);
```

A. 无限次　　B. 0 次　　C. 1 次　　D. 2 次

（2）下面程序的功能是把 316 表示为两个加数的和，使两个加数分别能被 13 和 11 整除，请选择填空。

```
#include <stdio.h>
int main()
{
    int i = 0, j, k;
```

初始化：
int y = 10;
变量 y 初始化为 10。

Do-While 循环：
do-while 循环会先执行循环体，再检查条件。

第一次迭代：
循环体：y--;
这是后置递减操作，y 从 10 减到 9（表达式的值为 10，但此处未使用，因此 y 变为 9）。
条件：--y
这是前置递减操作，y 从 9 减到 8，条件值为 8（非零，视为 true），继续循环。

第二次迭代：
循环体：y--;
y 从 8 减到 7。
条件：--y
y 从 7 减到 6，条件值为 6（true），继续循环。

第三次迭代：
循环体：y--;
y 从 6 减到 5。
条件：--y
y 从 5 减到 4，条件值为 4（true），继续循环。

第四次迭代：
循环体：y--;
y 从 4 减到 3。
条件：--y
y 从 3 减到 2，条件值为 2（true），继续循环。

第五次迭代：
循环体：y--;
y 从 2 减到 1。
条件：--y
y 从 1 减到 0，条件值为 0（false），循环结束。

循环结束后：
此时 y 的值为 0。
printf ("%d\n", y--);
打印当前 y 的值（0），然后 y 自减为 -1（后置递减，但不再使用）。

程序输出：
程序最终输出 0。

```
do {i++; k = 316 – 13*i; } while ( _____D_____ );
        j = k / 11;
        printf ( "316 = 13 * %d + 11 * %d", i, j);
        return 0;
    }
```

A. k/11　　B. k%11　　C. k/11==0　　D. k%11==0

（3）下面程序的运行结果是（ D ）

```
#include <stdio.h>
int main()
{
    int y = 10;
    do { y--; } while ( --y );
    printf ("%d\n", y--);
    return 0;
}
```

--y；判断时是 1 —— -1，没有终止循环的条件
不等于0的常量；　——恒真
原封不动的调试；　？？？
都是死循环！
为什么？？？

A. -1　　B. 1　　C. 8　　D. 0

（4）若运行以下程序时，从键盘输入 ADescriptor↙（↙表示回车），则下面程序的运行结果是（ C ）

C
D

```
#include <stdio.h>
int main()
{
    char c;
    int v0 = 0, v1 = 0, v2 = 0;
    do {
        switch ( c = getchar() ) {
            case 'a' : case 'A' :
            case 'e' : case 'E' :
            case 'i' : case 'I' :
            case 'o' : case 'O' :
            case 'u' : case 'U' : v1 += 1;
            default : v0 += 1; v2 += 1;
        }
    } while ( c != '\n' );
    printf ("v0 = %d, v1 = %d, v2 = %d\n", v0, v1, v2);
    return 0;
}
```

这段代码的功能是：
统计输入的字符（直到遇到换行符 \n 为止）：
v1：统计元音字母（包括大小写 a/A, e/E, i/I, o/O, u/U）出现的次数。
v0：统计所有字符（包括元音、非元音和换行符）出现的次数。
v2：和 v0 一样，统计所有字符出现的次数（即 v0 == v2）。
输出结果：v0, v1, v2 的值。

关键点解析
switch-case 的执行逻辑：
c = getchar() 读取一个字符，并进入 switch 判断。
如果字符是元音字母（如 'a' 或 'A'），则 v1 += 1。
default 语句一定会执行（无论是否匹配 case），所以 v0 和 v2 每次都会 +1。
因此：
v0 和 v2 的值始终相同（因为它们都在 default 里自增）。
v1 仅当输入是元音字母时增加。

A. v0=7, v1=4, v2=7　　B. v0=8, v1=4, v2=8
C. v0=11, v1=4, v2=11　　D. v0=12, v1=4, v2=12

（5）下面程序的运行结果是（ B ）

```
#include <stdio.h>
int main()
{
    int a = 1, b = 10;
    do
```

```
    { b -= a; a++; } while ( b-- < 0 );
    printf ("a = %d, b = %d\n", a, b);
    return 0;
}
```

A. a=3, b=11　　B. a=2, b=8　　C. a=1, b=-1　　D. a=4, b=9

这段代码的主要逻辑是：

初始化：
x = 0（初始值）
s = 0（**用于累加**）

while 循环条件：

!x != 0
先计算 !x（逻辑非），**再判断是否** != 0。

x = 0 —— !x = 1（true）
1 != 0 —— true（循环继续）

循环体：
s += ++x;
++x **先自增** x（x 变为 1），**然后加到** s（s = 1）。

再次检查循环条件：
x = 1 —— !x = 0（false）
0 != 0 —— false（循环终止）

输出 s：
s 的值是 1（因为只执行了一次 s += ++x）。

关键点
while(!x != 0) **等价于** while(x == 0)：
!x **是** x == 0 的逻辑非。
!x != 0 **相当于** (x == 0) != 0，**即** x == 0。

循环只执行一次：
第一次 x = 0，**条件成立**，进入循环。
++x **使** x = 1，下次循环条件 x == 0 **不成立**，退出循环。

运行结果
程序输出：
1

（6）设有程序段：

```
int k = 10;
while ( k = 0 ) k = k – 1;
```

则下面描述中正确的是（ C ）

A. while 循环执行 10 次　　B. 循环是无限循环
C. 循环体语句一次也不执行　　D. 循环体语句执行一次

（7）设有以下程序段：

```
int x = 0, s = 0;
while ( !x != 0) s += ++x;
printf ("%d", s);
```

则（B）。

A. 运行程序段后输出 0
B. 运行程序段后输出 1
C. 程序段中的控制表达式是非法的
D. 程序段执行无限次

2. while(!x != 0) **的逐步解析**
while **的条件是** !x != 0，我们需要拆解它的计算顺序：

(1) !x（逻辑非运算）
! 是逻辑非运算符，它的规则是：

如果 x **是** 0，!x **返回** 1（true）。

如果 x **是非** 0，!x **返回** 0（false）。

当前 x = 0，**所以** !x 的值是 1。

(2) !x != 0
上一步得到 !x = 1，现在计算 1 != 0：

1 != 0 **是** true（因为 1 **不等于** 0）。

所以 while(!x != 0) **等价于** while(1)（即循环继续）。

（8）语句 while(!E); 中的表达式!E 等价于（ D ）

A. E==0　　B. E!=1　　C. E!=0　　D. E==1

（9）下面程序段的运行结果为（ D ）

```
a = 1; b = 2; c = 2;
while (a < b < c) { t = a; a = b; b = t; c--; }
printf ("%d, %d, %d", a, b, c);
```

A. 1, 2, 0　　B. 2, 1, 0　　C. 1, 2, 1　　D. 2, 1, 1

（10）下面程序段的运行结果是（D）

```
x = y = 0;
while (x < 15 ) y++, x += ++y;
printf ("%d, %d", y, x);
```

A. 20, 7　　B. 6, 12　　C. 20, 8　　D. 8, 20

（11）对 for(表达式 1; ;表达式 3) 可理解为（ B ）

A. for(表达式 1; 0 ;表达式 3)
B. for(表达式 1; 1 ;表达式 3)
C. for(表达式 1; 表达式 1 ;表达式 3)
D. for(表达式 1; 表达式 3 ;表达式 3)

（12）以下程序段（ C
-14）

```
x = -1;
do { x = x * x; } while ( !x );
```

A. 是死循环　　B. 循环执行二次

C. 循环执行一次　　D. 有语法错误

（13）以下 for 循环的执行次数是（ C ）

```
for ( x = 0, y = 0; (y = 123) && (x < 4); x++ );
```

A. 是无限循环　　B. 循环次数不定

C. 4 次　　D. 3 次

（14）以下不是无限循环的语句为（　　）

A. for (y=0, x=1; x>++y; x=i++)　　i=x;

B. for (; ; x++=i);

C. while(1) {x++;}

D. for(i=10; ; i--) sum+=i;

（15）下面有关 for 循环的正确描述是（ D ）

A. for 循环只能用于循环次数已经确定的情况

B. for 循环是先执行循环体语句，后判断表达式

C. 在 for 循环中，不能用 break 语句跳出循环体

D. for 循环的循环体语句张红，可以包含多条语句，但必须用花括号括起来

（16）设有程序段：

```
t = 0;
while ( printf (”*”) ) { t++; if ( t < 3) break; }
```

下面描述正确的是（ ）

A. 其中循环控制表达式与 0 等价

B. 其中循环控制表达式与’0’等价

C. 其中循环控制表达式是不合法的

D. 以上说法都不对

whlie中表达式会执行，输出 * **且相当于恒真，**判断一次执行一次

（17）以下描述中正确的是（ D ）

A. 由于 do-while 循环中循环体语句只能是一条可执行语句，所以循环体内不能使用复合语句

B. do-while 循环由 do 开始，用 while 结束，在 while(表达式)后面不能写分号

C. 在 do-while 循环体中，一定要有能使 while 后面表达式的值变为零（“假”）的操作

D. do-while 循环中，根据情况可以省略 while

（18）有以下程序段：

```
int n = 0, p;
do { scanf (”%d”, &p); n++; } while ( p != 12345 && n < 3 );
```

此处 do-while 循环的结束条件是（ D ）

A. P 的值不等于 12345 并且 n 的值小于 3

B. P 的值等于 12345 并且 n 的值大于等于 3

C. P 的值不等于 12345 或者 n 的值小于 3

D. P 的值等于 12345 或者 n 的值大于等于 3

（19）以下程序的输出结果是（D）

```
#include <stdio.h>
int main()
```

```
{
    int a, b;
    for ( a = 1, b = 1; a <= 100; a++)
    {
        if ( b >= 10 ) break;
        if ( b % 3 == 1) { b += 3; continue; }
    }
    printf (”%d\n”, a);
    return 0;
 }
```

A. 101　　B. 6　　C. 5　　D. 4

（20）以下程序中，while 循环的循环次数是（ D ）

```
int i = 0;
while ( i < 1 )
{
      if ( i < 1 ) continue;
      if ( i == 5 ) break;
      i++;
 }
```

A. 1　　B. 10

C. 6　　D. 死循环，不能确定次数

3、程序填空题

（1）下面程序段是从键盘输入的字符中统计数字字符的个数，用换行符结束循环。请填空。

```
int n = 0, c;
c = getchar();
while ( ____c!='\n'____ )
{
    if ( ____c>='0'&&c<='9'_____ )      n++;
    c = getchar();
}
```

（2）下面程序的功能是：输出 100 以内能被 3 整除且个位数为 6 的所有整数，请填空。

```
#include <stdio.h>
int main( )
{
    int i, j;
    for ( i = 0; ____j<100_____; i++)
    {
        j = i * 10 + 6;
        if ( _j%3!=0_ ) continue;
        printf (”%d”, j);
     }
     return 0;
```

```
}
```

（3）下面程序的功能是：求 Fibonacci 数列的前 40 个数，并按照 4 列 1 行输出。请填空。

```
#include <stdio.h>
int main( )
{
    long f1 = 1, f2 = 1;
    int i;
    for ( i = 1; i <= 20; i++)
    {
        printf ("%-12d%-12d", f1, f2);
        if ( i%2==0 ) printf ("\n");
        f1 = _____f1+f2_____;
        f2 = _____f2+f1_____;
     }
     return 0;
}
```

这里是一次显示两个数，所以以二为除数